FiveM

FiveM resources, frameworks, txAdmin, permissions, databases and troubleshooting.

- Check ESX Legacy dependencies and resource startup
- Check QBCore resources, database readiness and boot order
- Configure FiveM endpoints and resource start order
- Connect FiveM resources to a private database with oxmysql
- Diagnose FiveM joining problems on the assigned TCP and UDP port
- Give FiveM staff limited txAdmin access
- Install a FiveM resource and check its manifest dependencies
- Resolve FiveM Asset Escrow entitlement and key errors
- Set up a FiveM server licence key in Cfx Portal
- Set up pma-voice and test proximity, radio and calls
- Test FiveM ACE permissions without granting full command access
- Trace FiveM script errors and measure resource performance
- Update FiveM listing details and artwork using current Portal settings
- Keep FiveM resource secrets out of replicated convars
- Test a FiveM game build change before using it on production
- Fix FiveM resource filename case problems after moving to Linux
- Schedule txAdmin restarts with warnings and a save checkpoint
- Collect txAdmin logs for support without exposing player data
- Create an isolated FiveM staging copy with a separate database
- Update a Git-managed FiveM resource while preserving local settings

Check ESX Legacy dependencies and resource startup

ESX Legacy is a collection of resources, not just a folder called `es_extended`. Its core version, supporting libraries, database schema and additional scripts must agree. This guide helps check an existing installation or prepare a separate test setup. It is not permission to run a fresh-install recipe over a live roleplay server.

Identify the installation you have

Record the ESX release or commit, FXServer artifact and database adapter release. Read the official core repository and the configuration shipped with that release. Save a backup of all resources and configuration plus a matching database export before changing core files. Keep a list of custom modifications, especially inventory, identity, multicharacter and jobs integrations.

If you are comparing a txAdmin recipe, inspect its actions before running it. Recipes can create or import databases and replace files. Use a new isolated directory and database for a trial deployment. Existing characters, money and inventories are not safe merely because a recipe says its installation succeeded.

Check dependencies before add-ons

The official configuration checked in November 2026 starts `oxmysql`, then `esx_lib`, then `es_extended`, followed by the remaining core resources. Treat that as release-specific reference, not a guarantee that every older ESX package has identical dependencies. Inspect your release's manifest and documentation.

```
# Simplified dependency order, not a complete configuration.
ensure oxmysql
```

```
ensure esx_lib
ensure es_extended
ensure [core]
```

Keep required settings, including the private database connection and the release's OneSync configuration, before resource startup. Preserve exact resource names and case. Do not rename a library to make its folder look tidier: imports and exports may reference that name. If a category also contains a resource already ensured above, review the effective startup carefully rather than adding more duplicate lines.

Read the first boot error

- **Database connection failure:** establish the adapter's connection before investigating character scripts.
- **Missing table or column:** compare installed schema and migration history with the matching ESX release. Do not recreate the production database.
- **Missing export or library:** confirm the named dependency started successfully and is the expected version.
- **Old shared-object event errors:** identify the add-on's supported ESX integration. Use the author's current migration instructions; do not paste a guessed replacement into every resource.

Disable a newly added optional script only during planned maintenance, leaving the known working core unchanged. Restarting ESX beneath running jobs and inventory scripts can leave inconsistent state. Prefer a controlled full restart after saving player data.

Test the complete player path

Join with a test account, reach character selection or creation, spawn and perform one ordinary inventory or job action. Disconnect normally, reconnect and confirm the expected character state persists. Test existing characters as well as new ones: a schema mismatch may affect only old records. Keep tests small and avoid commands that give money or items to real players.

For support, provide the first redacted startup error, exact ESX/core versions, adapter version and recent resource changes. Remove licence keys, connection strings and player identifiers. A functioning database connection alone does not prove ESX character loading or an add-on's compatibility.

Related guides

- Connect oxmysql to your database
- Check manifests and resource dependencies

Further reading

- ESX Legacy official core repository
- ESX reference configuration
- ESX core resource manifest

Looking for another game? Browse game server setup and troubleshooting guides.

Check QBCore resources, database readiness and boot order

A working QBCore server needs compatible core, character, inventory, database and supporting resources. Uploading `qb-core` by itself will not produce a complete roleplay server. Start by identifying which official recipe or release your installation follows, then compare the installed resources to that reference before changing anything.

Preserve the live installation

Take a resource and configuration backup together with a matching database export. Record custom resources and their supported QBCore versions. A fresh txAdmin deployment can import SQL and replace configuration, so use a separate test directory and database for experiments. Do not rerun a recipe against existing player data to resolve an isolated missing export.

The official upstream repository URLs currently resolve to the `qbcore-fivem` organisation; older links may use `qbcore-framework`. Follow the official project's redirects and recipe references rather than trusting a download purely because its filename says QBCore. Record actual commits or releases; mixed snapshots can disagree about exports and item definitions.

Make dependency readiness visible

Use your recipe's full configuration as the reference. For diagnosis, the essential relationship is database settings first, the database adapter ready, then core, then dependent resources. A deliberately simplified example is:

```
# Supply your private connection setting before this block.
ensure oxmysql
```

```
ensure qb-core
ensure [qb]
```

This does not replace the recipe's standalone, voice, map and default resources. Review those groups too. Ensure the adapter is present once under its required resource name and examine its first startup message. In a managed container the database can be on another endpoint; do not substitute `localhost` for the address supplied with your database.

The official recipe checked for this guide pins an oxmysql download rather than treating any fork's newest build as interchangeable. Follow the dependency reference for your chosen recipe version, and test upgrades separately. Do not assume a modern core will support every older `qb-` resource unchanged.

Trace errors from the beginning

A missing `GetCoreObject` export can result from a failed or incompatible core, a renamed folder or an outdated add-on. Check those causes before editing calls globally. Unknown item or job errors need comparison with the core's shared definitions and the add-on's instructions. A missing SQL table points to schema compatibility, not necessarily a broken network connection.

Keep framework administration separate from txAdmin roles. Do not copy owner-level command grants from a sample into ordinary player permissions. Fix the particular resource's access requirements without making every staff member a god or granting all console commands.

Verify characters and persistence

After a planned restart, join with a test account and complete character selection, spawn and a normal inventory action. Disconnect cleanly and reconnect to confirm persistence. Test an existing character after any approved migration; successful new-character creation can hide problems with older rows. Check the client F8 log as well as the server console when a loading screen hangs.

Send support the first error, core and add-on versions, database adapter release and the exact action that fails. Redact connection strings and identifiers. Restore files and database together if an approved change requires rollback, rather than mixing a previous schema with new resource files.

Related guides

- Configure the database adapter

- Test pma-voice proximity and radio

Further reading

- QBCore official core repository
- QBCore official txAdmin recipe
- QBCore recipe configuration
- QBCore recipe dependency downloads

Looking for another game? Browse game server setup and troubleshooting guides.

Configure FiveM endpoints and resource start order

FiveM can read several configuration files, especially when txAdmin profiles or `exec` includes are used. Before changing endpoints or resource order, identify the file the running profile executes. Editing an unused copy explains many cases where a saved setting appears to do nothing. Keep a private backup of the active file and any included files.

Use the allocation you were given

The familiar FiveM port 30120 is an example, not permission to use it on every service. Match the game port assigned to your server. The two lines below are a template: replace `ASSIGNED_GAME_PORT` with the same numeric allocation in both lines before saving.

```
endpoint_add_tcp "0.0.0.0:ASSIGNED_GAME_PORT"
endpoint_add_udp "0.0.0.0:ASSIGNED_GAME_PORT"
```

A wildcard bind address means interfaces available inside the server environment. It is not a join address and does not open a provider firewall. In a container, the public IP may not exist directly inside the container, so forcing FXServer to bind that public address can fail. Keep the host-provided bind arrangement unless support confirms a change. Current Enhanced documentation limits each endpoint type to one; do not copy old multi-endpoint configurations without checking your branch.

Build an explicit startup chain

Settings required during resource initialisation belong before the relevant `ensure` lines. For a database-backed custom resource, a simplified chain is:

```
# Connection setting is private; supply real values separately.
set mysql_connection_string "REPLACE_WITH_PRIVATE_CONNECTION_STRING"
ensure oxmysql
ensure example_framework
ensure example_job
```

This illustrates dependency order, not a complete ESX or QBCore installation. Use the recipe matching your framework release. Start the database adapter before the framework, then framework-dependent jobs and interfaces. Avoid relying on the order of unrelated resources inside a bracketed category. If a specific resource needs another, make the dependency explicit in its manifest when you control the code.

Find duplicate or conflicting settings

Inspect the main file and every `exec` include for repeated endpoint, key and connection assignments. Check whether the hosting startup process writes values automatically. Preserve a single authoritative location for each setting. Do not paste an entire internet sample over your current configuration: examples can change ports, database names, game builds, permissions and resource sets all at once.

Use `ensure` carefully. Issuing it for an already running resource restarts that resource, which can interrupt players or discard unsaved script state. Use a planned full restart for changes to a framework's boot sequence rather than restarting core components underneath dependent scripts.

Verify from startup through joining

After a maintenance restart, watch for binding errors and the first failed dependency. Confirm database readiness, then framework readiness, then the dependent feature. Test a real external connection to the assigned public address. If the server starts correctly but cannot be reached, collect the address, port and timestamp for network diagnosis; changing resource order will not repair a blocked UDP allocation.

Keep RCON disabled unless you have a specific secured management requirement. Do not publish your full configuration in a support thread: remove licence keys, database passwords, bot tokens and webhooks from any excerpt.

Related guides

- Diagnose external FiveM connections
- Install resources and dependencies

Further reading

- Cfx server command reference
- Cfx default server recipe
- FiveM Legacy and Enhanced differences

Looking for another game? Browse game server setup and troubleshooting guides.

Connect FiveM resources to a private database with oxmysql

oxmysql is the database adapter used by many FiveM resources. It does not create a managed database account or automatically install your framework's schema. You need the database hostname, port, database name, username and password supplied for your service. Use an account limited to your own game database, never a shared database administrator account.

Choose a release honestly

Overextended and CommunityOx have both had archival changes. CommunityOx's organisation and oxmysql repository are archived as of this guide's November 2026 check. Its installation documentation remains useful historical reference; it is not evidence of active maintenance. Use the release and download origin currently recommended by your framework's official recipe, record the exact version, and check compatibility before switching forks. Do not choose a similarly named download site because it ranks first in search.

Before changing the adapter, back up the resource files and export the matching database. Preserve the existing working connection configuration privately. If an upgrade needs migrations, read those migrations before running them. Do not import a starter schema over production tables.

Set the connection before resources start

Install the built release as the resource name `oxmysql`. In the active server configuration, use the adapter's supported connection format. This template deliberately contains placeholders and cannot connect until you replace them:

```
set mysql_connection_string
"user=DB_USER;password=DB_PASSWORD;host=DB_HOST;port=DB_PORT;database=DB_NAME"
```

```
ensure oxmysql
ensure example_framework
```

Keep this a private `set`, not `setr` or a public server-listing value. The database address must be reachable from the game container. `localhost` means that container's own network context; it is usually wrong for a separately hosted database. Use the endpoint CloudExa supplies, including a non-default port if one was assigned.

Reserved characters need the format and escaping supported by your selected adapter release. Never weaken a password or publish a connection string to get past parsing trouble. If a secret was exposed in a log or screenshot, rotate it and update the active configuration.

Read the first failure

- **Timeout or refused connection:** check the exact hostname and port, service availability and whether the game server's source is allowed. Ask support to check the narrow network path; opening the database to everyone is unnecessary.
- **Access denied:** check credentials and the database account's permitted source host. The same password working from your home PC does not prove the container is authorised.
- **Unknown table or column:** the connection probably reached SQL; check the resource's schema and migration version.
- **Missing export:** verify resource name, release compatibility and startup order before changing SQL.

Verify persistence

After startup reports a successful connection, test one normal framework action that writes data. Reconnect and verify that change remains. Use a read-only query through your authorised database tools if needed. Avoid enabling full query logging permanently: queries can contain personal information and create heavy logs. Save the first redacted error, adapter release and framework version if you need support.

Related guides

- Check ESX Legacy startup
- Check QBCore startup

Further reading

- Historical oxmysql project installation reference
- CommunityOx archive status
- oxmysql repository and release history
- MariaDB account host matching

Looking for another game? Browse game server setup and troubleshooting guides.

Diagnose FiveM joining problems on the assigned TCP and UDP port

A FiveM server can appear online in a panel while players cannot join. The process being running does not prove that its public TCP and UDP paths work. Test the address assigned to the service, then compare the exact failure with startup logs before changing firewall rules or resource files.

Confirm the connection target

Copy the server's assigned public IP and game port from its connection details. Do not assume the port is 30120, and do not use the txAdmin web address or database endpoint. If a hostname is involved, check that it resolves to the intended address. A website proxied through an ordinary HTTP CDN does not automatically proxy FiveM game traffic.

Use a device outside the server environment for the join test. In your FiveM client's F8 console, replace the placeholders below with your allocation:

```
connect PUBLIC_GAME_IP:ASSIGNED_GAME_PORT
```

Record the time and full error. Test a second permitted network if possible, such as another player's connection, to distinguish a single-client problem from an all-player outage. Do not ask players to disable their security software broadly.

Check the server side without assuming root access

Inspect the FXServer console for an endpoint binding failure, licence failure or repeated crash. Confirm the active configuration binds both TCP and UDP on the assigned game port. A container may bind `0.0.0.0` internally while the host maps the public address; forcing a public IP that is absent inside the container can prevent startup.

If you own a VPS and are authorised to manage its firewall, check only the assigned game port and both protocols. On a managed game service, ask CloudExa to verify allocation and host firewall mapping. Do not install a firewall inside a container, flush host rules, or open every port as a troubleshooting shortcut.

Understand what a web check proves

From your own computer, an optional HTTP check against the assigned game port can help distinguish basic TCP reachability. Replace the placeholders before running this in a local terminal:

```
curl --connect-timeout 5 --max-time 10 "http://PUBLIC_GAME_IP:ASSIGNED_GAME_PORT/info.json"
```

A valid response shows that this TCP/HTTP path answered at that time. It does not prove UDP traffic reaches FXServer, that a client can finish joining, or that framework character loading works. Conversely, an HTTP response can be intentionally restricted or proxied in some deployments; interpret it alongside the actual client test. A generic TCP port checker cannot certify UDP.

Separate joining from gameplay failures

If the client reaches resource downloads and then fails on one file, inspect that resource or download path. If it reaches character selection and hangs, look at framework and database errors. If direct connection works but the listing is missing, check Cfx authentication and listing configuration separately; repeatedly changing the game port can create a second problem.

For support, provide the service ID, public address and port, test time, whether both tested networks failed, and the redacted console error. Keep licence keys and authentication tokens private. Leave txAdmin and RCON restricted: exposing management interfaces is not required to let players join the game.

Related guides

- Check endpoint binding in server.cfg

- Verify Cfx server authentication

Further reading

- Cfx server endpoints reference
- Cfx client console commands
- Cfx proxy and connection architecture

Looking for another game? Browse game server setup and troubleshooting guides.

Give FiveM staff limited txAdmin access

txAdmin has its own staff accounts and permissions. A CloudExa game-panel subuser, an ESX group, a QBCore permission and a txAdmin administrator are separate access systems. Granting one does not automatically grant the others. Use a separate named account for each staff member so actions can be traced and access removed individually.

Check which control surface you have

txAdmin is bundled with FXServer, but your service must actually run it and provide an authorised management address. Use only the address supplied for your service. A normal game connection port is not automatically the txAdmin web port. If no txAdmin address is supplied, ask support about availability rather than exposing extra ports or changing startup arguments yourself.

Before changing an established staff setup, save a private backup of the txAdmin profile and access configuration through the tools available to you. Keep it away from public downloads: account and authentication information is sensitive. If you are also changing framework permissions, take the corresponding files and database backup before that separate change.

Create a role around real duties

Open txAdmin's Admin Manager as the owner or an administrator authorised to manage accounts. Give a moderator the specific player actions their job requires. For example, viewing the player list and warnings does not require permission to change server settings or execute arbitrary console commands. Select actions individually using the installed version's permission descriptions.

In particular, `console.write`, `server.cfg.editor`, `commands.resources` and `control.server` are operational powers. Treat `manage.admins` as permission to grant access to others. The `all_permissions` role is owner-level access, not a convenient default for ordinary staff. Protect the master account and avoid distributing its password.

Check web and in-game access separately

1. Have the staff member sign in using their own account in a separate browser session. Confirm the permitted pages and actions work.
2. Check an action outside their role is unavailable. Do not run a destructive action just to test its denial; inspect the controls or use a harmless, controlled test.
3. If they need the in-game menu, attach their correct Cfx.re or Discord identifier to their txAdmin account. A display name alone is not the identifier.
4. After joining, test `/tx` or `/txadmin`. If permissions were recently changed, the documented `/txAdmin-reauth` command can retry authentication.

Remove access cleanly

When someone leaves, remove their txAdmin account and separately review game-panel, database and framework permissions. If they knew shared secrets, rotate those secrets through the appropriate service. Review txAdmin action logs around unexplained restarts or configuration changes before assuming the hypervisor failed.

If menu authentication fails, record the exact error and account identifier type privately. Do not turn off source-IP checks or other authentication safeguards as a first fix. A reverse proxy or unusual hosting layout needs support to check its trusted network path. Successful staff access means both the allowed action works and owner-only actions remain restricted.

Related guides

- Understand ACE permissions and principals
- Collect useful resource error evidence

Further reading

- txAdmin permission system
- txAdmin in-game menu
- txAdmin project and setup

Looking for another game? Browse game server setup and troubleshooting guides.

Install a FiveM resource and check its manifest dependencies

A downloaded resource must have its manifest at the correct directory level before FXServer can discover it. Installing a script also means meeting its dependencies, selecting the correct framework variant and preserving any existing configuration. A resource appearing in the console is only the first check; its actual game feature must work too.

Check the download before uploading

Use the author's release or authorised purchase download. A repository source archive can lack built web-interface files included in a release. Read the version requirements and configuration instructions. Before replacing an existing resource, back up its folder and configuration; if it stores player data, export the matching database as well. Keep that rollback pair together.

Extract into a resource directory beneath your server's `resources` folder. For example, `resources/[local]/example_resource/fxmanifest.lua` is sensible. An accidental extra level such as `example_resource/example_resource/fxmanifest.lua` can prevent discovery. Bracketed folders group resources; the resource name remains `example_resource`. Keep author-required names and letter case, particularly on Linux.

Read the manifest

Modern resources normally use `fxmanifest.lua`. The following is an illustrative manifest for a resource you are writing, not a replacement for a purchased script's manifest:

```
fx_version 'cerulean'
game 'gta5'
```

```
client_script 'client.lua'
server_script 'server.lua'
dependency 'example_library'
```

The declared script paths must exist. Dependencies can include another resource and runtime requirements, so read the author's notes rather than deleting a dependency line to force startup. Client and server files run in different contexts; database credentials belong in server configuration, never a client script or shared web asset. Do not modify protected escrow files.

Start and test deliberately

Install the documented dependency release first. In the FXServer console, with a maintenance window if players are online, discover newly added resources and then start the required components:

```
refresh
ensure example_library
ensure example_resource
```

Replace those example names. `refresh` rescans resources; it does not install missing libraries. `ensure` starts a stopped resource or restarts one already running, so do not use it casually on a live core framework. Observe the first error rather than repeatedly issuing the command. Missing exports often mean a dependency did not start, was renamed, or exposes a different API version.

Make the result survive restart

Add the confirmed order to the active configuration, then perform a planned restart. Join using a test account and exercise the feature, including any save-and-rejoin behaviour. A clean startup with no visible feature may point to framework configuration or permissions rather than networking. Add one new resource at a time so the last change remains easy to identify.

If installation fails, restore the previous folder and configuration during maintenance. Do not import a fresh schema into a live database or remove player tables as a troubleshooting shortcut. Share the resource name, release, first error and dependency versions with support, redacting secrets and player identifiers.

Related guides

- Order resources in server.cfg
- Trace resource errors and performance

Further reading

- Cfx resource manifest reference
- Cfx introduction to resources
- Cfx server commands

Looking for another game? Browse game server setup and troubleshooting guides.

Resolve FiveM Asset Escrow entitlement and key errors

Asset Escrow protects paid FiveM resources and checks whether the server is entitled to run them. An entitlement error is not usually a reason to reinstall the game server or replace its database. Check account ownership, the active server licence and the original download first. Keep the exact first console error because missing entitlement and damaged resource files are different failures.

Check the account relationship

Sign in to the official Cfx Portal using the account that should own the asset. Confirm the purchase or subscription entitlement exists and is valid. A Tebex payment receipt is useful evidence, but purchasing through the wrong linked Cfx account can still leave the server's owner without entitlement. Ask the seller through their official support route to resolve a legitimate account issue.

Now identify the Cfx account owning the licence key actually used by this FXServer instance. An old key left after an ownership change can authenticate the server while failing the purchased asset's entitlement check. Do not borrow another person's key or share your key with a script seller in a public support channel. Older instructions may call the licence site Keymaster; use the current Portal links.

Check the deployed package

Back up the installed resource and its configuration before replacing files. If the resource stores persistent data, export the matching database as well. Compare the installed files with a new authorised download of the same compatible version. Extract the archive fully: hidden entitlement-related package files must not be skipped by your upload method. Preserve the directory structure and author-required resource name.

Use the seller's built distribution rather than a random mirror or a source archive missing its packaged assets. Do not edit encrypted code, strip escrow metadata or try tools claiming to decrypt the resource. Changing `escrow_ignore` locally does not grant ownership or make protected vendor files editable.

Separate licence failure from resource failure

1. Check startup for a server licence authentication error before the resource's own message. Resolve that first if present.
2. For an entitlement refusal, compare the asset-owning account with the active server key's account and check any subscription expiry.
3. For missing or unreadable protected files, recheck the original authorised package and completed upload.
4. For a dependency, export or SQL error after the entitlement check passes, use the resource's normal installation guide instead of repeatedly regenerating keys.

Where account changes or an updated package require a restart, arrange a short maintenance window and use the normal server controls. Avoid repeated live restarts of economy or inventory resources. Retest the actual feature with a test player, rather than treating a single successful start message as complete validation.

Escalate with safe evidence

Provide the seller with the asset name, purchase reference, account identity through their private channel, resource release and exact redacted error. Provide CloudExa with the service ID and time if you suspect the wrong configuration or blocked licence-service connectivity. Never include the full server key, database password or player database in a public ticket attachment.

Do not assume an asset is transferable just because the server moved providers or owners. Follow the current Cfx and seller rules for the particular purchase. Once fixed, record which account owns the server key and assets so a later handover does not recreate the same problem.

Related guides

- Check the active Cfx server licence
- Verify resource files and dependencies

Further reading

- Cfx Asset Escrow documentation
- Cfx Portal
- Cfx resource manifest escrow directives
- Cfx server licence setup

Looking for another game? Browse game server setup and troubleshooting guides.

Set up a FiveM server licence key in Cfx Portal

A FiveM server needs a valid Cfx.re server licence key. Create and manage it in **Cfx Portal** using the account that should own the server. Older tutorials call this Keymaster; follow the current Portal workflow rather than an old screenshot. The licence authenticates FXServer. It does not replace your CloudExa panel password or grant another account permission to use purchased resources.

Before changing the key

Note your assigned public connection address and the configuration file your service actually starts. A txAdmin profile can use a different server-data folder from the one you first uploaded. Save a private copy of the active configuration. For an established roleplay server, also take a file backup and a matching database export before making wider setup changes.

1. Sign in to the official Portal with your own Cfx account. Use its server registration/licence workflow and provide the details it currently requests. Where an IP is requested, use the assigned server public IP, not your home address, a database host or a website proxy.
2. Keep the generated licence key private. Do not paste it into chat, screenshots, public GitHub repositories or a client-side resource.
3. Identify whether your hosting startup configuration supplies the key. If it does, update that setting; a manual file change may otherwise be overwritten when the service starts.

Apply it to the active configuration

If your service reads the key from `server.cfg`, the setting has this form. Replace the entire placeholder and keep only the intended active key assignment:

```
sv_licenseKey "REPLACE_WITH_YOUR_CFX_SERVER_KEY"
```

Do not use a replicated setting such as `setr` for secrets. Check included configuration files as well as the main file: a later assignment can replace an earlier one. A key belonging to a previous owner can explain why basic authentication works while a purchased asset reports that entitlement is missing.

Prove the change worked

Arrange a short restart through your normal game-server controls. Watch startup from the beginning and distinguish a licence authentication error from a later resource error. If authentication succeeds but a script fails, do not regenerate the key repeatedly; troubleshoot the named script instead. Test joining with your assigned game address after FXServer finishes starting.

Current Cfx recipes also direct server-listing metadata to Portal. If a name or description is being managed there, editing a legacy convar may not change the listing. Check the instructions for your actual FXServer branch and Portal account before assuming the saved file was ignored.

If authentication still fails

Record the exact error, time, FXServer artifact version and whether you recently moved IP or ownership. Confirm the active profile contains the new key without revealing its value. Ask CloudExa to check outbound connectivity if the error describes reaching Cfx services; do not disable security checks or borrow another customer's key. Rotate any key exposed publicly, then update the one active startup source.

Related guides

- Configure FiveM endpoints and resource order
- Resolve Asset Escrow entitlement errors

Further reading

- Cfx server setup with txAdmin
- Cfx Portal
- Cfx default server recipe

Looking for another game? Browse game server setup and troubleshooting guides.

Set up pma-voice and test proximity, radio and calls

pma-voice uses FiveM's internal Mumble voice system. Proximity voice, a framework radio item and a phone-call interface are related but separate features. Test proximity first; a broken radio script does not by itself show that the game server's voice transport is blocked.

Choose a compatible build

Read the author's release notes and the requirements of your phone, radio and framework integrations. The main branch can contain breaking changes before a stable release. Current author notes distinguish calls from older phone-named configuration and state fields, so do not combine an old phone script with new settings blindly. Pin the selected release or commit for a reproducible test.

Back up the current voice resources and configuration before changing them. If your radio or phone integration stores persistent data, also export its matching database. Schedule a short maintenance window, because restarting voice while people are connected can interrupt conversations and complicate the test.

Run one voice implementation

Install the resource with the required name `pma-voice` and remove conflicting startup entries for other voice systems after preserving their configuration. Disable voice features embedded in other menus if the author requires it. Multiple scripts setting proximity and Mumble audio distances can override each other even when both claim to have started successfully.

This example shows a modest native-audio configuration. Check the options against your installed release and place settings before its startup line:

```
setr voice_useNativeAudio true
setr voice_useSendingRangeOnly true
ensure pma-voice
```

Do not enable several alternative audio modes at once. Start the framework's radio and phone integration only after their documented prerequisites. Do not invent an external Mumble IP or open arbitrary voice ports: the normal integrated setup should be checked against the service's assigned networking first. An external voice-server design needs its own approved configuration.

Use two real clients

1. Have two players join and confirm their operating-system input device and FiveM voice settings are correct. Check push-to-talk and microphone privacy permissions.
2. Stand near each other and speak, then move beyond the selected proximity range. Verify both directions; one-way voice can be a client device or script-state issue.
3. Change proximity mode using the installed key binding. Existing players may retain their personal key settings even after a default changes; inspect FiveM key bindings rather than assuming a new default replaced them.
4. Test a shared radio channel separately, then a phone call separately. Record which test failed and any client or server error.

Keep diagnosis narrow

If proximity works but radio does not, inspect the radio resource's integration, item access and channel permissions. If only phone calls fail, compare that script's API expectations with the installed pma-voice version. Avoid restarting the entire framework as a first reaction. For all-player silence after an update, review the last configuration change and competing voice resources.

Disable temporary verbose logging once evidence is captured. Share the voice release, framework integration versions, failing test and redacted error with support. Do not record private conversations as a diagnostic shortcut; a controlled consented microphone test is enough.

Related guides

- Check QBCore dependencies
- Diagnose the assigned game connection

Further reading

- pma-voice author documentation

- pma-voice release history
- pma-radio author project
- Cfx voice scripting reference

Looking for another game? Browse game server setup and troubleshooting guides.

Test FiveM ACE permissions without granting full command access

FiveM's ACE system answers whether a principal may use an object. A principal can be an identifier, group or resource; an object is a permission name checked by a command or script. Inheritance connects principals. This is separate from txAdmin's staff permissions and may also be separate from your framework's job or group system.

Start with the resource's documented object

Find the exact ACE name the resource checks. A script checking `examplemenu.open` will not necessarily use a guessed group called moderator. Likewise, a command only uses the built-in restricted-command check when registered that way; ACE configuration cannot secure an event the script exposes without checking permission. Ask the author when their access model is unclear.

Back up your permission configuration and included files before editing. If a framework stores access assignments in its database, export the matching database too. Work with a test account, and record any temporary console commands so you can undo them. Console changes should not be mistaken for durable configuration.

Use a harmless scoped test

The following uses an invented permission object solely to demonstrate ACL behaviour. Run it in the FXServer console as the owner. It does not grant an actual administrative command:

```
add_ace group.help_demo cloudexa.helpcheck allow
test_ace group.help_demo cloudexa.helpcheck
remove_ace group.help_demo cloudexa.helpcheck allow
```

```
test_ace group.help_demo cloudexa.helpcheck
```

The first test should show permission; the second should not unless another rule supplies it. If the result differs, inspect inheritance and existing rules before adding more grants. This proves the ACL mechanism, not that any installed script uses the invented object.

Connect a real user carefully

For a real permission, use the player's verified stable identifier and the author's exact object. A conceptual assignment looks like this; replace the identifier placeholder privately before use:

```
add_principal identifier.license:REPLACE_WITH_VERIFIED_ID group.example_staff
add_ace group.example_staff examplemenu.open allow
```

This grants only the named object's access, subject to other applicable rules. Do not copy an identifier from a public screenshot or assume the player's current numeric server ID is permanent. Test the intended object against that principal with `test_ace`, then test the actual feature with the player. Keep screenshots and shared logs free of full player identifiers.

Understand inheritance before widening access

Adding a principal to a group can inherit everything that group already allows. A broad parent can silently give more power than the two lines you just added suggest. Explicit deny rules can also explain a refusal. Avoid granting the whole `command` namespace, owner groups or permission-management commands simply to silence one error.

Persist the smallest working rules in the intended configuration file and verify them after a planned restart. Remove a temporary relationship with `remove_principal` using the same child and parent arguments. Removing one relationship does not remove permissions inherited through another path. Retest both the desired feature and an unrelated restricted feature, then document who has access and why.

Related guides

- Set limited txAdmin staff access

- Check active server configuration

Further reading

- Cfx ACL command reference
- Cfx RegisterCommand native
- txAdmin separate permission system

Looking for another game? Browse game server setup and troubleshooting guides.

Trace FiveM script errors and measure resource performance

A red error line, hitch warning and low client FPS describe different problems. Start with the earliest relevant error and the action that triggers it. Repeated later errors can be consequences of one failed dependency, so collecting a short startup sequence is more useful than restarting resources until the console becomes quiet.

Capture a useful reproduction

Record the resource name, installed release, FXServer artifact, time and player action. Check both the server console and the affected player's F8 console. A server-side SQL exception and a missing client texture need different fixes. Keep the first error and its stack trace together; the final line alone often omits the function or file that failed.

Before editing or updating a resource, back up its files and configuration. For persistent frameworks, export the matching database too. Use a test copy where possible. Do not hide an error by removing permission checks, suppressing every exception or replacing player tables with a fresh schema.

Classify the error before changing code

- **Missing export or dependency:** verify the referenced resource exists, started successfully and matches the expected API version.
- **Nil value:** inspect the named variable's setup and prerequisite state; a missing player or configuration value can trigger it.
- **SQL error:** distinguish connection failure from a missing table, incompatible query or schema version.

- **Client-only error:** reproduce with another client and check asset paths, case and the author's supported game build.

Change one thing at a time and repeat the same test. A resource restart can disrupt dependent scripts, so schedule maintenance for core changes rather than restarting the whole framework under connected players.

Measure on the correct side

On a test client's F8 console, `resmon true` can show per-resource client timing and memory; use `resmon false` afterwards. Its availability depends on branch and developer-mode policy. Legacy instructions using a client launch argument do not automatically apply to Enhanced, whose documentation describes a different developer-mode setup. Do not enable development access on a production server merely to bypass a denied command.

Client resmon does not measure the server's entire CPU load or database latency. Compare readings under the same player location and workload. A resource that is cheap while idle may become expensive in a busy shop or during an event.

Take a bounded profile

The Cfx profiler works in the appropriate server or client console. For a short authorised capture while reproducing the issue:

```
profiler record 500
profiler status
profiler saveJSON help-capture.json
```

Wait for recording to finish before saving. The count is frames, not seconds, so duration varies with workload. The output location belongs to the process that recorded it; a client capture is not saved in your hosted server's files. Ask support if server-side output is outside your available file access.

Review the capture for costly functions and compare before and after a single change. Share profiles privately because paths and script details can be sensitive. Keep raw timing evidence when asking an author for optimisation help; adding RAM does not automatically repair a blocking loop or an inefficient database query.

Related guides

- Inspect resource dependencies
- Distinguish database failure types

Further reading

- Cfx profiler guide
- Cfx scripting performance reference
- Cfx client console reference
- Enhanced developer-mode differences

Looking for another game? Browse game server setup and troubleshooting guides.

Update FiveM listing details and artwork using current Portal settings

Use this guide when your FiveM listing has outdated text or artwork while players can otherwise connect. The current Cfx documentation directs several presentation settings to Portal, so older tutorials that only edit `server.cfg` may no longer describe your server's behaviour. Keep your licence key and account recovery codes private throughout this change.

Identify the right project

Record your server's connection address, current public name and existing artwork. Save the active configuration and take a screenshot of the listing. Sign in to Cfx Portal with the authorised account and identify the project associated with this service. If you cannot establish ownership, ask the account owner to make the change; do not create replacement keys simply to edit a label.

The current server command reference marks `sv_hostname`, `sets sv_projectName`, `sets sv_projectDesc` and `load_server_icon` as deprecated in favour of Portal. Follow the controls available for your project. An older runtime may retain legacy behaviour, so record the installed artifact and confirm its supported path with support before removing working configuration.

Make a small presentation change

1. Prepare a recognisable project name and a plain sentence describing the community. Keep keyword lists, payment information and private staff details out of the name.
2. Update only the intended Portal fields, preserving unrelated ownership and service settings. Keep a copy of the previous values for rollback.
3. For a configuration-managed banner, use `sets banner_detail "https://YOUR-IMAGE-HOST/banner-v2.png"` with your own public URL. The placeholder is not an upload destination.
4. Prepare the documented 73:8 banner ratio; Cfx currently recommends 2560 by 280 pixels. Keep important text legible when scaled down and use artwork you have

permission to publish.
5. Open the image address in a private browser window. It must display the image without a login, expiring token or sharing-page button. Keep the previous image hosted until verification is complete.

Verify from outside your account

Refresh the public listing and check it from a second browser or player's client. Confirm the name, description, icon and banner independently. The documented `banner_detail` is used for both listing details and the connection modal; `banner_connecting` is deprecated and has no effect in the current reference.

Join through the unchanged address and verify the expected server still loads. Presentation edits should not require changing endpoints, database credentials or resource start order. If the image remains stale, check the public URL and distinguish an image cache from a configuration value that never applied. Avoid rapid unrelated edits while waiting for listing refreshes.

Rollback and support evidence

Restore the saved text or previous image URL if the result is unreadable or belongs to the wrong project. Supply support with the public listing URL, artifact version, changed field and timestamp if values refuse to update. Redact authentication information. A listing that looks correct does not establish access control; maintain your normal whitelist and permission rules separately.

Related help

- Set up a FiveM server licence key in Cfx Portal
- Diagnose FiveM joining problems on the assigned TCP and UDP port

Keep FiveM resource secrets out of replicated convars

A FiveM resource may read configuration through convars, but the command used to define a value determines who can see it. Use this audit before adding database credentials, webhooks or API keys to configuration. You need permission to edit your own server files and access to the resource's official setup instructions.

Understand the exposure boundary

Cfx distinguishes standard convars set with `set`, client-replicated values set with `setr` and public server-information values set with `sets`. Server-information values appear in the listing and `info.json`. Replicated values can be read by client scripts. Neither is an appropriate place for a private credential.

A standard convar is not a password vault. Server-side resources and authorised administrators can still access server configuration, and careless logging can expose it. Use a server-only setting only when the resource documents that method; some resources use their own configuration files or environment variables instead.

Audit without spreading the secret

1. Back up the relevant configuration privately. Do not paste it into a public validator or screenshot the entire file.
2. List credential names, not their values: database connection, webhook, third-party token and any resource-specific private keys. Find every place each is assigned, including included configuration files.
3. Compare the setting with the resource's instructions. If it requires a server-only convar, use its documented name with `set`, not `setr` or `sets`.
4. Keep harmless client preferences separate. A language or interface colour may legitimately be replicated; replacing every `setr` with `set` can break a resource that

needs client access.
5. Check client scripts and downloadable resource files for hard-coded copies. A value hidden from the listing can still leak if included in a client bundle.

Handle a value that was already public

Assume an exposed key may have been copied. Restrict or revoke it with the issuing service, create a replacement with only the required permissions, and update the documented server-only location. For database credentials, coordinate the change so dependent resources receive the new value together. Do not simply hide the old text and leave the old key valid indefinitely.

Make this change during a planned window where the resource requires restart. Check its startup output for successful authentication while avoiding debug modes that print connection strings. Test one ordinary resource action. A successful boot alone does not prove that a background integration can authenticate.

Verify and recover

Inspect your own public server-information response and client-visible settings without publishing the results. Confirm the secret's name and value are absent where they should be private. Review recent logs for accidental disclosure and restrict any retained diagnostic copies.

If functionality fails, recheck quoting, configuration order and the resource's expected setting type. Roll back harmless structural changes if necessary, but never restore a compromised credential as the long-term fix. Current Cfx convar rules describe the transport boundary; the individual resource remains responsible for reading, storing and logging the value safely.

Related help

- Connect FiveM resources to a private database with oxmysql
- Install a FiveM resource and check its manifest dependencies
- Test FiveM ACE permissions without granting full command access

Test a FiveM game build change before using it on production

A resource asking for another GTA game build is not necessarily asking for a newer FXServer artifact. Those are separate compatibility decisions. Test a game-build change before applying it to your live roleplay community, especially when vehicles, interiors or scripts depend on a particular content set.

Record the current combination

Save your effective startup configuration and record the FXServer artifact, game generation, enforced game-build setting, framework release and affected resource versions. Read the resource author's stated requirements. If a purchase page and bundled README disagree, resolve that discrepancy before changing production.

The current Cfx command reference says `sv_enforceGameBuild` takes effect at startup, not during runtime. It also distinguishes FiveM for GTAV Enhanced from Legacy: do not assume an older Legacy build listed in the table is available on Enhanced. Select from the current documentation for your actual runtime instead of copying an old “latest build” number.

Prepare a separate test

1. Use a staging service with separate ports, credentials and database. Copy only an authorised checkpoint; keep production data untouched.
2. Verify the current game-build configuration works in staging first. Otherwise a pre-existing resource error may be blamed on the proposed change.
3. Read the affected resources' manifests and release notes. The resource manifest reference describes runtime constraints that a resource can declare; those declarations complement, rather than replace, the author's support matrix.

4. Change only the game-build setting to the supported target. Keep artifact updates, framework migrations and unrelated script changes out of this experiment.
5. Stop and start staging normally. Preserve the first startup output, including any resource refusal or unsupported-build message.

Test more than joining

Join using the client generation you intend to support. Check character loading, the affected vehicle or map asset, a representative inventory action and one persistence action after reconnecting. If the change was required for a particular resource, exercise that resource's actual feature rather than using an unrelated spawn as proof.

Ask a second authorised tester to repeat the relevant scenario. Record exact failures, client generation and timestamps. Do not recommend that every player delete caches or reinstall GTA before establishing whether the server is requesting a supported build. Client download time also belongs in your maintenance expectations.

Promote or reverse the change

If the checks pass, schedule production maintenance, take a fresh consistent checkpoint and apply the same single setting. Repeat the short acceptance checklist after startup. Keep a staff member available for the first normal joining sessions.

If staging fails, restore its previous build setting and compatible checkpoint as needed. A resource or framework update that changed stored data needs its own rollback plan; restoring one startup number does not reverse a database migration. Preserve the logs and ask the resource maintainer about the exact supported combination. Recheck this guide's upstream references whenever your runtime generation changes, because a stable build number is not a universal compatibility promise.

Related help

- Install a FiveM resource and check its manifest dependencies
- Trace FiveM script errors and measure resource performance
- Schedule game server backups and check retention

Fix FiveM resource filename case problems after moving to Linux

A FiveM resource that worked on one machine may fail after a move to a case-sensitive filesystem. A reference to `Config.lua` is not interchangeable with an actual file named `config.lua` there. Investigate the exact missing path rather than reinstalling the whole framework or clearing unrelated server data.

Preserve and locate the failure

Save the first file-not-found error and the resource name. Download a private copy of the affected resource before editing. Record its release and compare its directory layout with the author's original package. Keep configuration containing credentials out of public issue reports.

Check whether the archive added an extra outer folder. A resource manifest buried inside a second copy of the resource directory is a packaging problem, not necessarily a case problem. First confirm which folder the server actually starts and whether it contains the expected `fxmanifest.lua` .

Compare literal paths

1. Open the manifest and compare each reported script, file and UI path with the real filename, character for character. The Cfx manifest reference explains how those files are declared.
2. Check directory names as well as the final extension. A correct `menu.js` still cannot be found through the wrong `HTML/` directory name.
3. For a browser interface, inspect references inside its HTML and stylesheet files too. A manifest can load the page successfully while an image or JavaScript import uses different case.
4. Look for two files that differ only by case. Compare their contents and the author's release before deciding which one is intended. Do not overwrite one with the other on your local

computer.
5. Correct the smallest proven mismatch in a test copy. Prefer the author's documented naming convention and avoid broad search-and-replace across unrelated resources.

Move the corrected copy carefully

If your editor or file manager does not register a case-only rename, rename through a temporary unique name, then the intended final spelling. Do this in a stopped test copy so the resource cannot briefly load an incomplete path. Git's configuration reference documents filesystem-related case handling; do not change repository case options blindly to conceal a genuine filename conflict.

Upload the complete corrected resource and compare the resulting filenames in the panel. Keep the previous version in your private backup, outside the active resource search path. Avoid leaving duplicate working copies with confusing names in folders that the server automatically starts.

Verify the affected feature

Start the resource on staging and check that the original error is gone. Open the specific menu, load the relevant asset or run the action that previously failed. Look for new missing-file errors rather than accepting an empty interface as success.

If the correction fails, restore the saved resource and configuration together, then reassess packaging, build output and declared dependencies. Some packages require a build step and do not ship their generated web assets. An encrypted or vendor-managed resource may require an upstream fix; do not edit protected files. Apply the verified minimal correction to production only during an appropriate maintenance window.

Related help

- Install a FiveM resource and check its manifest dependencies
- Trace FiveM script errors and measure resource performance

Schedule txAdmin restarts with warnings and a save checkpoint

A scheduled restart should give players warning and leave a known recovery point. txAdmin supports scheduled restarts, but the existence of a countdown does not prove that a framework saved character, inventory or economy data. Configure the schedule and persistence checks as separate responsibilities.

Choose one scheduler

Check the game panel, txAdmin and any framework resources for existing restart jobs. Record their times and timezones. Select one system to own the restart; disable only a duplicate job you understand and have permission to change. Two schedulers can interrupt each other or restart during an unfinished backup.

Confirm the installed txAdmin version and your administrator permissions. The upstream project lists scheduled restarts and warnings as features. Exact setting labels vary by release. Use the schedule controls shown in your installation rather than pasting an outdated configuration block.

Build the maintenance sequence

1. Select a quiet time and publish it with an explicit timezone. Check daylight-saving changes when your staff and hosting region use different clocks.
2. Enable player warnings. Tell players when to stop trades, purchases and other actions that could remain incomplete at shutdown.
3. Read your framework's documented save procedure. Confirm which resources persist on shutdown and which require an explicit supported checkpoint. There is no universal command that safely saves every FiveM economy.
4. Schedule a consistent backup with enough time to finish before the restart. Where a stopped backup is required, make that maintenance process explicit rather than pretending it can run safely inside a short countdown.

5. Configure one initial restart and have a staff member observe it. Do not schedule frequent automatic restarts to mask an unresolved resource crash.

Use events only with reviewed code

Developers can inspect txAdmin's documented events. The scheduled-restart event supplies remaining time, while the shutdown event supplies a delay before termination. These are notifications, not a guarantee that an asynchronous database write completed.

If your framework integrates with those events, test the exact installed integration on staging. Do not paste a generic event handler that stops every resource in arbitrary order. A handler that starts a long export immediately before termination may produce a partial backup while appearing to run normally.

Verify the first scheduled restart

Check that warnings appear to a normal player, the server stops once and it starts at the expected time. Compare the console and txAdmin action log with the schedule. Reconnect and inspect a known test character's position, inventory and other agreed persistent state. Verify the backup completed independently of the successful restart.

If the timing or saved state is wrong, suspend the new schedule and use a supervised maintenance window while investigating. Restore the previous scheduler setting if necessary, retaining logs and backups. Do not repeatedly roll the database backward while players reconnect. Agree any recovery point first and keep the failed run's timestamps for framework support.

Related help

- Give FiveM staff limited txAdmin access
- Schedule game server backups and check retention
- Trace FiveM script errors and measure resource performance

Collect txAdmin logs for support without exposing player data

txAdmin logs can explain a failed resource start, unexpected restart or staff action, but a full export may expose player identifiers, chat and typed commands. Send support a focused copy while preserving the original evidence privately. You need authorised access to the service's logs; do not change permissions across the host to obtain them.

Identify the right time and profile

Write down the incident time, timezone, service name and exact symptom. Include whether a player saw it once or repeatedly. Compare the panel clock with your own clock before choosing an excerpt. A five-minute timezone mistake can make a useful log look unrelated.

The official log reference distinguishes admin events, FXServer console output and server activity. Choose the stream that answers your question. An administrative restart belongs in a different investigation from a resource stack trace or a player chat dispute.

Find the configured log location

Use the panel's available log viewer or file manager. Older layouts commonly place logs below a txData profile, but check the actual installation. The current environment documentation describes `TXHOST_DATA_PATH` and notes changes from older convars. On hosted services, ask support when the data path is outside your accessible files; do not edit startup settings to make it visible.

1. Confirm the profile belongs to the affected service. A familiar folder name alone is not enough if several test profiles exist.
2. Copy the relevant log files to a private local folder. Keep original names and timestamps, and record which period each covers.
3. Extract the incident window with a few minutes of context before the first failure. Include the first error, not only the last repeated line.

4. Make a second copy for redaction. Keep the untouched original restricted in case support later needs to verify sequence or correlation.
5. Review the whole excerpt for credentials, webhooks, database URLs, session tokens, chat and player addresses. Replace sensitive values consistently, such as `[PLAYER-A]`, while preserving useful error structure.

Send a reproducible report

Include the resource version, artifact and txAdmin version, steps that triggered the issue and the expected result. Attach the redacted excerpt through your private support ticket. Avoid publicly accessible paste sites for raw console logs, and do not send an entire txData directory when a short excerpt answers the question.

If you suspect a staff-account compromise, preserve the full originals and tell support privately before changing retention. Deleting log files does not necessarily remove related records elsewhere, and it can destroy evidence of the original sequence.

Verify and retain appropriately

Open the submitted attachment to confirm it is readable and covers the incident. Check that redaction did not remove the resource name, time or stack-trace line needed to investigate. Keep a record of the support ticket and original file location.

This is an export workflow, not a retention-policy change. If logs rotate before incidents can be investigated, agree suitable retention with support and your privacy obligations. Do not increase every log to unlimited retention or enable verbose credential-bearing debug output indefinitely. Remove only your temporary working copies when the case and your agreed retention policy permit it.

Related help

- Give FiveM staff limited txAdmin access
- Trace FiveM script errors and measure resource performance

Create an isolated FiveM staging copy with a separate database

A staging server is useful only when a test cannot damage the live community. A copied resource folder that still connects to production MySQL is not isolated. Use this checklist before testing framework migrations, paid resources or administrator tools. You need a separate allocated service and authority to copy the relevant data.

Define what may leave production

List the files, database tables and integrations required for the test. Use synthetic accounts where possible. If realistic data is necessary, use an authorised consistent export and minimise personal information. Do not copy raw player records to a developer's public repository or expose the test server as an unrestricted download source.

Record the production artifact, framework and resource versions. Capture a baseline backup before making the copy. Request separate connection allocations and a separate database with its own restricted user. Keep the production server online or stopped according to the documented consistency requirements of your export process, not according to convenience.

Remove live connections before first start

1. Upload the copied files while staging is stopped. Keep its startup configuration separate and confirm you are editing the test service.
2. Replace every database connection with the isolated database address and credentials. Review included configuration files and environment settings as well as the main configuration.

3. Replace or disable payment, reward, email, Discord and other webhook integrations. A test action should not deliver real purchases or notify every member of your community.
4. Give staging separate administrator credentials and appropriately authorised Cfx settings.

   Keep private values server-side; the convar reference explains why replicated or public information convars are unsuitable for secrets.
5. Confirm the txAdmin data path belongs to staging. Current txAdmin environment settings differ from older configuration methods, so have support verify host-managed paths instead of overriding them blindly.

Prove isolation

Before starting the framework, have the database administrator confirm the test account can access only the intended test database. Start staging and inspect its connection messages without printing passwords. Make a harmless test record through the application, then verify it appears only in staging. Do not test isolation by attempting a destructive production query.

Restrict joining with your supported access controls and give staff the test address directly. Do not rely on a friendly “TEST” name as access control. Keep a written check that no production-facing integration fired during the test. If an integration cannot be safely disabled, postpone that feature's test until an appropriate sandbox is available.

Run and reverse one change

Take a staging checkpoint, apply one update and follow its acceptance checklist. Record both failures and successful persistence checks. Restoring staging should use a matching code, configuration and database checkpoint together.

If isolation fails, stop staging immediately, preserve its logs and assess which external actions occurred before restarting it. Rotate credentials if copies escaped their intended boundary. Never promote the staging database back over production after players have continued playing; production rollout requires a fresh backup and the verified migration procedure applied to current data.

Related help

- Connect FiveM resources to a private database with oxmysql
- Install a FiveM resource and check its manifest dependencies
- Schedule game server backups and check retention

Update a Git-managed FiveM resource while preserving local settings

Blindly replacing a Git-managed resource can erase local settings or leave a mixture of old and new files. Prepare the update on your own administrator workstation or an authorised development environment, then upload a complete reviewed release. This guide does not require a root shell on the game node or permission to run Git inside the game container.

Record the current resource

Write down the upstream repository, installed release or commit, resource folder name and local configuration files. Download a private copy of the deployed directory, including files ignored by Git. Record whether the release requires generated web assets, dependencies or a database migration.

A repository history is not a complete operational backup. Credentials and runtime data should normally remain outside public source control, while ignored files may contain exactly the settings you need to recover. Do not upload your deployment folder to a public repository to make this update easier.

Inspect changes locally

1. Open the local repository used for development. Run `git status --short` to identify tracked edits and untracked files. The Git status reference describes these categories.
2. Review `git diff` and `git diff --cached` privately. These inspect different sets of changes; check both before assuming the working copy matches the installed release.
3. Preserve local settings in an external private backup and a reviewed development branch where appropriate. Never commit passwords simply to obtain a clean working tree.
4. Obtain the target release from the resource's authorised upstream project. Compare its release notes and file changes with the installed revision using your Git client's comparison view. The official diff documentation explains comparison output.

5. Reapply only intentional local changes to the target copy. Merge configuration by meaning rather than replacing a new schema with the entire old file. Keep a note of each retained customisation.

Package and test the target release

Follow the author's build instructions when the source repository does not include deployable web assets. Do not treat a GitHub source archive as equivalent to a published compiled release. Review the resource manifest and check that referenced scripts and packaged files exist in the final upload.

Test the prepared directory on an isolated service with a separate database. Confirm startup, the resource's main action and persistence after reconnecting. If a migration is required, test it on a copied database first and confirm the author's downgrade limitations. A code revert cannot automatically undo altered tables.

Deploy and retain recovery copies

Schedule a production window and take a fresh consistent checkpoint. Stop the affected service as its framework requires, deploy the complete prepared release and repeat your acceptance checks. Avoid leaving another active copy of the resource in an automatically started folder.

If checks fail, stop further writes and use the matching pre-update code, configuration and database recovery plan. Do not run destructive Git reset or clean commands against the live deployment. Keep the old private package, target revision and test record until the update is accepted and your backup retention policy allows cleanup.

Related help

- Install a FiveM resource and check its manifest dependencies
- Trace FiveM script errors and measure resource performance
- Connect FiveM resources to a private database with oxmysql